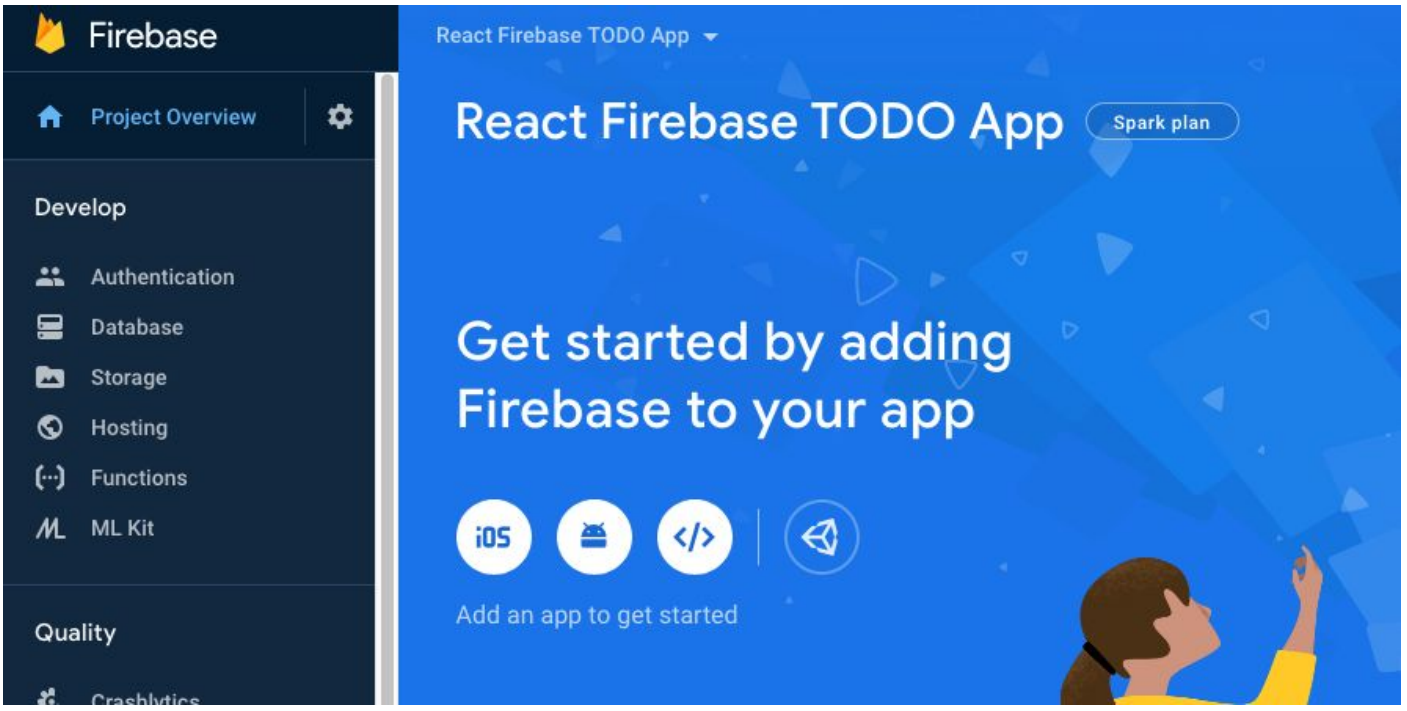
React und Firebase

In dieser Übung werden wir eine React Anwendung erstellen, die die Daten in Googles Firestore ablegt und den Zugriff über Firebase Authentication regelt.

Firebase Projekt erstellen

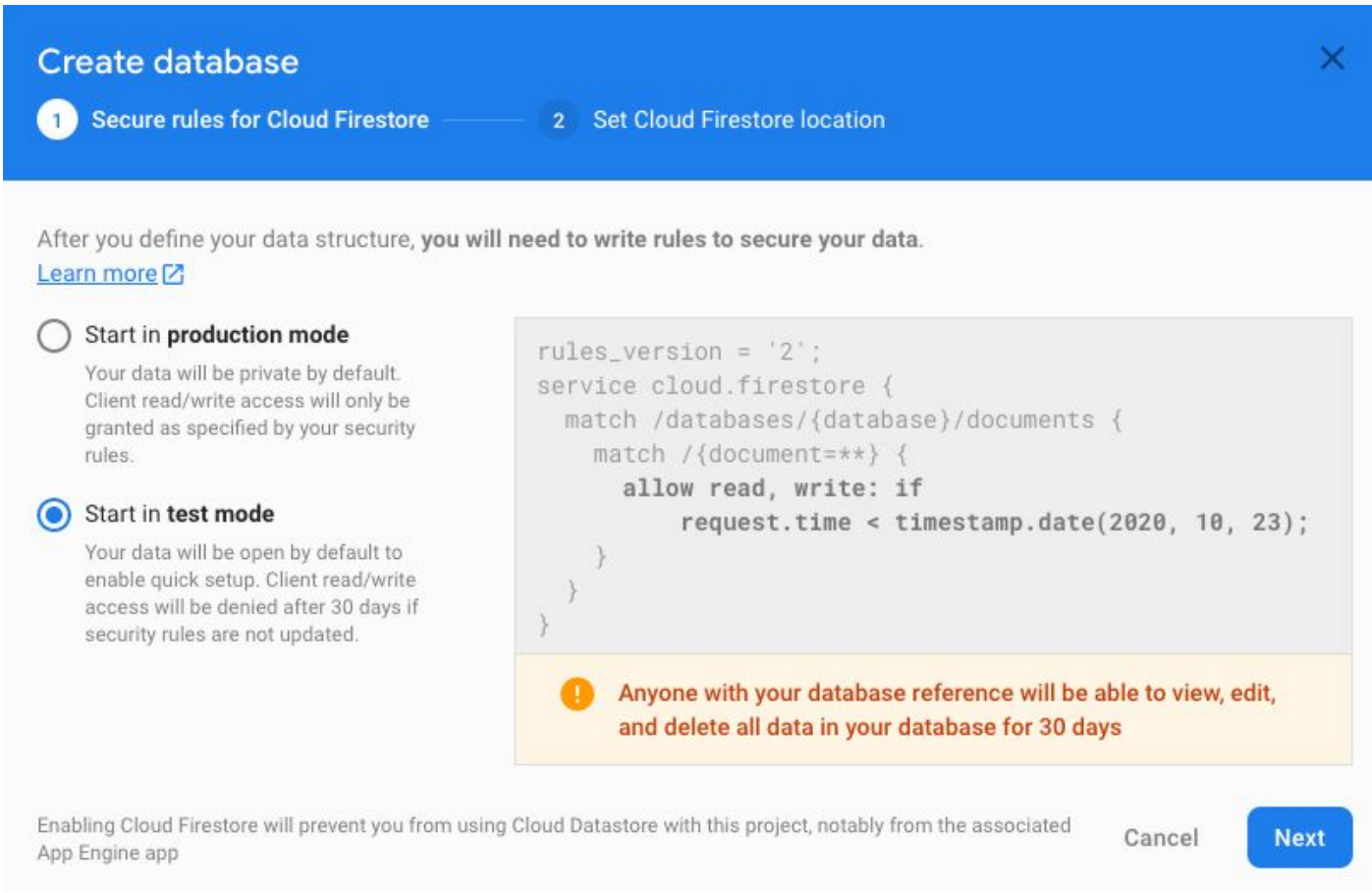
Als erstes benötigen Sie ein Firebase-Projekt. Wahlweise können Sie das Projekt auch mit anderen Teilnehmern teilen, indem Sie ihnen auch Zugriff auf das Projekt geben (Project Settings -> Users and Permissions). Das Projekt erstellen und konfigurieren Sie folgendermassen:

1. Öffnen Sie die Firebase console und erstellen Sie ein neues Projekt. Alle Optionen, um Daten mit Google zu teilen und zu Analytics, können Sie getrost deaktivieren.
2. Das Projekt wird anschliessend erstellt, und Sie landen in der Project Overview, wo Sie eine Web-App (</> Icon) hinzufügen müssen:



3. Vergeben Sie nochmals einen Namen und aktivieren Sie auch gleich das Hosting. Die weiteren Schritte für das Hosting können Sie ignorieren, wir werden diese weiter unten erledigen, bzw. hat die Vorlage dies zum Teil schon gemacht. Am Ende landen Sie wieder in der Konsole
4. Unter **Develop** → **Authentication** in der Firebase-Konsole aktivieren Sie im **Sign-In Method** Tab mindestens **Google** oder auch weitere Provider, wie zum Beispiel **Email/Password**.
5. Unter **Cloud Firestore** wählen Sie **Create Database** und zum Anfang starten wir im *test mode*. Um die Absicherung werden wir uns später kümmern. Welche Location Sie im nächsten Schritt wählen

ist ganz egal.



React Projektvorlage klonen

Die Vorlage für das Projekt finden Sie auf GitHub. Erstellen Sie davon einen lokalen Klon:

```
git clone https://github.com/IFS-Web/HSR.WED3.Exercise.React.Firebase.git
```

Jetzt müssen Sie nur noch die ganzen Dependencies des Projekts installieren und die Vorlage mit Ihrem Firebase-Projekt verbinden:

1. `npm install`
2. `npx firebase login` loggt Sie ein und erstellt ein Projekt
3. `npx firebase projects:list` zeigt das oben erstellte Projekt ID anzeigen lassen:

   `yet-another-todo-list` | `yet-another-todo-list-782ae`

6. Verbinden Sie das Firebase-Projekt mit der Vorlage:
   `npx firebase use yet-another-todo-list-782ae`
7. Als nächstes muss das Firebase SDK konfiguriert werden:
   Mit `npx firebase apps:sdkconfig web` erhalten Sie die Konfigurationsparameter, welche Sie in die Datei `src/config.js` übernehmen können. Kopieren Sie nur die **Attribute** in das exportierte Objekt.
8. `npm run build`

9. Mit `npx firebase deploy` können Sie die Vorlage nun deployen und sich danach in der deployten App anmelden.

Während der Entwicklung können Sie mit `npm start` einen lokalen Server starten.

React Firestore

In der Klasse `TodosContainer` finden Sie eine Implementation der TODO-Liste mit React-State. Diesen State wollen wir nun aber im Firestore persistieren. Ergänzen Sie als Erstes `firebase.js` um einen Export für die Datenbank (mit dem `db.settings(...)` Aufruf wird eine Warnung, die für uns nicht relevant ist, abgestellt):

```
export const db = firebase.firestore();
db.settings({
  timestampsInSnapshots: true
});
```

Jetzt können Sie in `TodosContainer` die `db` importieren und verwenden. Für die Abfrage der TODOs wollen wir über Snapshot-Updates automatisch über Änderungen benachrichtigt werden.

```
componentDidMount() {
  db.collection("todos")
    .onSnapshot(this.handleOnNext, this.handleOnError);
}
```

Die TODOs, die Sie über `handleOnNext` erhalten, können Sie wieder im State der Komponente ablegen. Die übrigen Callbacks (`handleToggle`, `handleDelete`, `handleCreate`) hingegen arbeiten nur mit dem Firestore, über die Snapshot-Updates gelangen die neuen TODOs dann wieder in den State und werden in den Views angezeigt.

Firestore Security Rules einrichten

Im Moment können alle an der Firebase-Instanz angemeldet Benutzer die TODOs aller Anwender sehen. Als nächster Schritt ist es also nötig, nur die TODOs des jeweils angemeldeten Benutzers zu zeigen. Dazu speichern Sie in den TODOs die User-ID des angemeldet Benutzers ab:

```
  db.collection("todos").add({
    text,
    checked: false,
    userId: auth.currentUser.uid
  });
```

Natürlich muss auch noch die Query angepasst werden:

```
componentDidMount() {
  db.collection("todos")
    .where("userId", "==", auth.currentUser.uid)
    .onSnapshot(this.handleOnNext, this.handleOnError);
}
```

So werden zwar nur noch die TODOs des Benutzers abgefragt, allerdings wurde noch keine wirkliche Sicherheit gewonnen. Bei der Erstellung des Firestores sind wir im *test* Modus gestartet, nun sollen aber nur a) angemeldete Benutzer b) ihre eigenen TODOs sehen. Das erreichen wir mit der folgenden Regel, die Sie in der Datei `firestore.rules` anpassen können:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /todos/{document=**} {
     allow read, update, delete: if request.auth.uid == resource.data.userId;
     allow create: if request.auth.uid != null;
    }
  }
}
```

Ein erneutes `npm run build` und `npx firebase deploy` übernimmt die Einstellungen. In der Firebase-Konsole sehen Sie die Rules dann unter **Database** -> **Rules**.

Beispiellösung

Die Beispiellösung finden Sie auch auf GitHub, im Master-Branch: Diff um von der Vorlage zur Lösung zu kommen.